CSCI 3753: Operating Systems

Week 6
October 3rd, 2025

University of Colorado **Boulder**

Announcements

- PA4 due Sunday at midnight!
- Quiz 6 due today at midnight
- Interview grading should done by end of day today!
- As always recitation materials are stored here

Recap

- PA0,1,2,3 ✅
- Problem Set 1 ✅
- PA4 🔁
- Introduced:
  - Shared queues
  - Mutual exclusion

On-Going

- PA4
  - Create a shared array
- PA5
  - Create a DNS resolver ⚠️
- PA6
  - Combine these to create a multi-threaded DNS resolver

On-Going

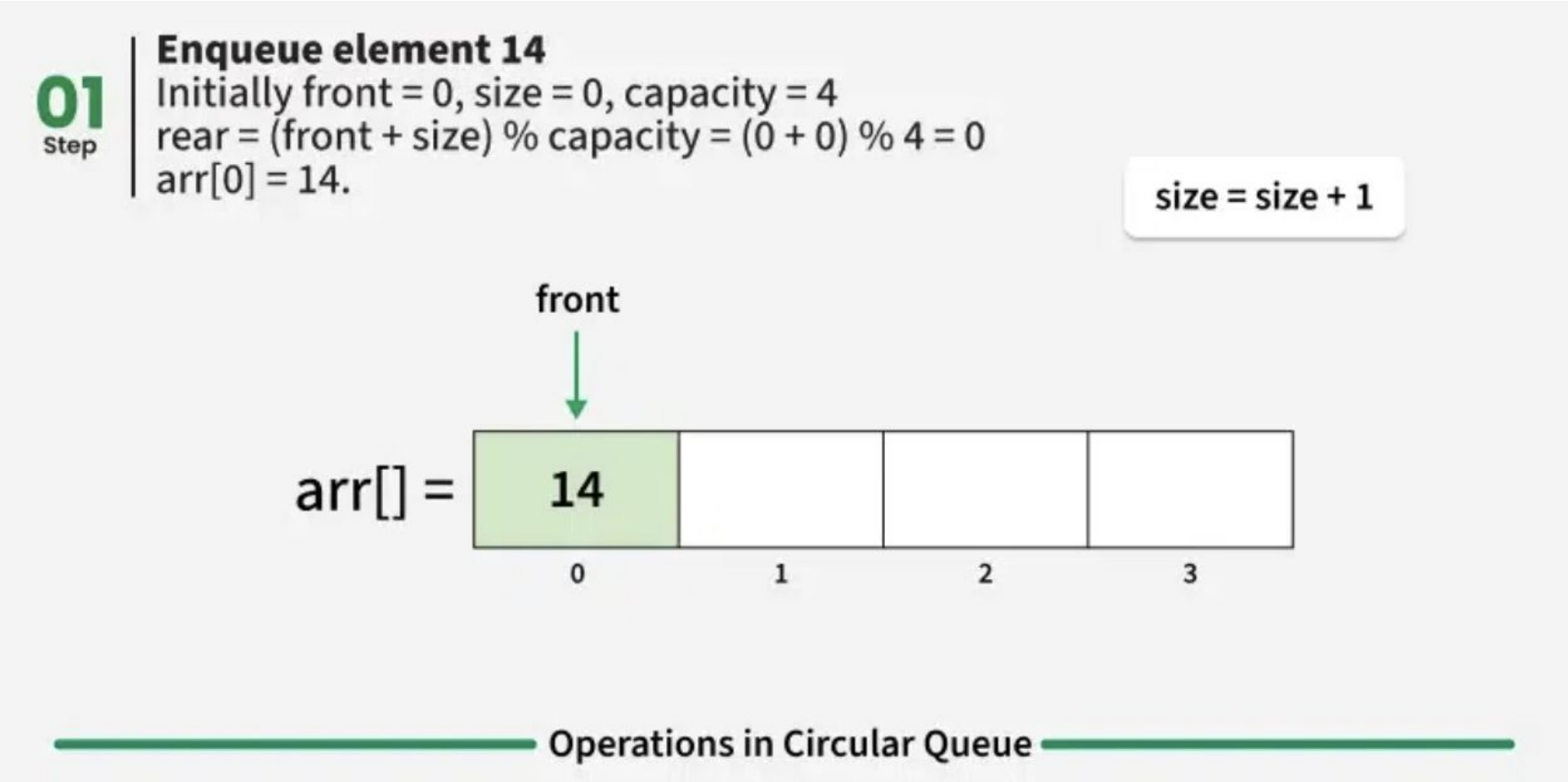
- PA4
  - Create a shared array
- PA5
  - Create a DNS resolver ⚠️
- PA6
  - Combine these to create a multi-threaded DNS resolver

Bounded Buffer Problem!!

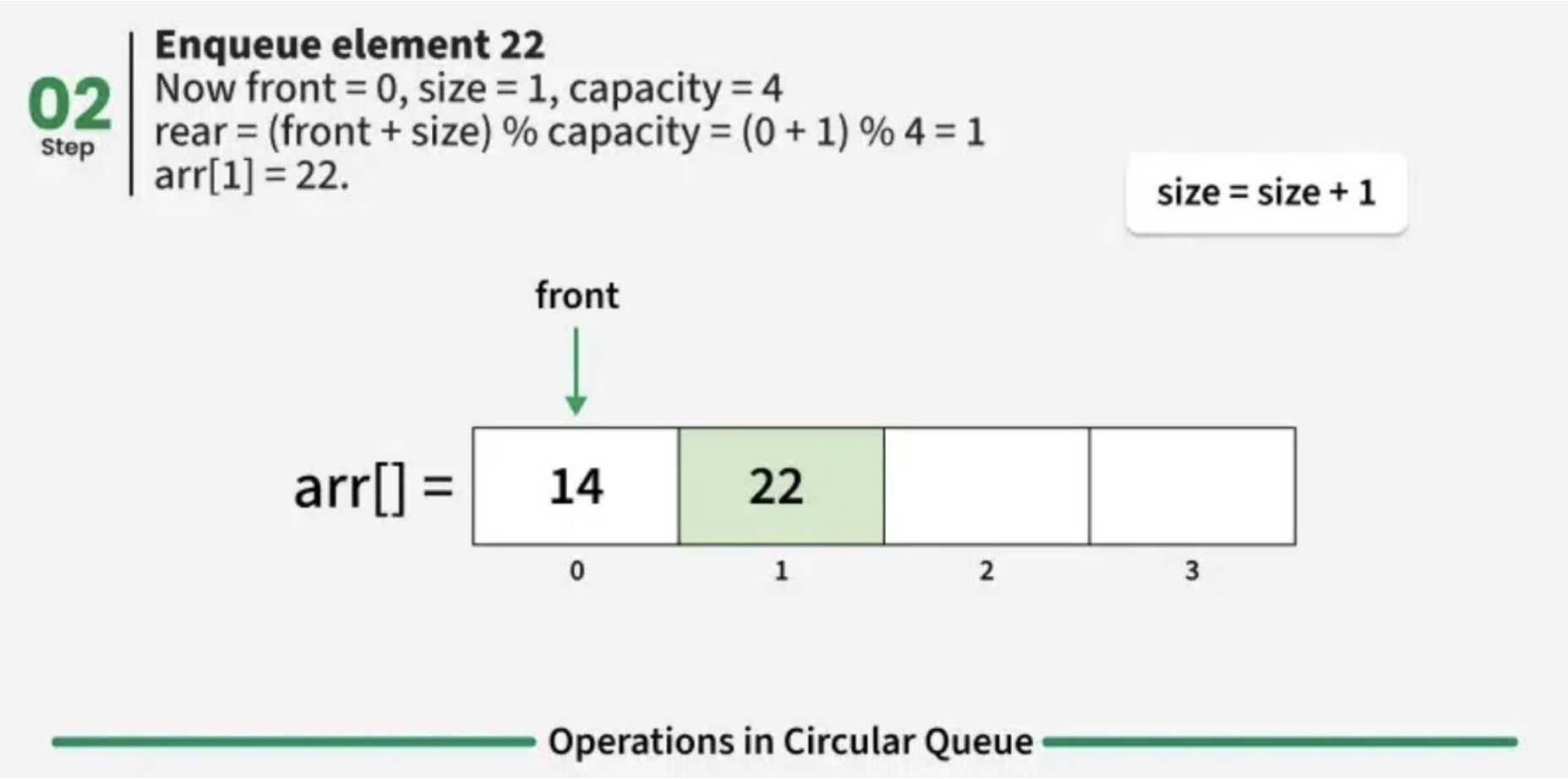
On-Going

- PA4
  - Create a shared array
    - Must be:
      - thread safe!
      - built on top of one (or more) contiguous, linear memory arrays
    - Could be:
      - FIFO
      - LIFO
      - Circular queue
    - Cannot have:
      - linked lists, dictionaries, trees, other pre-built data structures in C
  - Turn in both .h (header) and .c files for your shared array

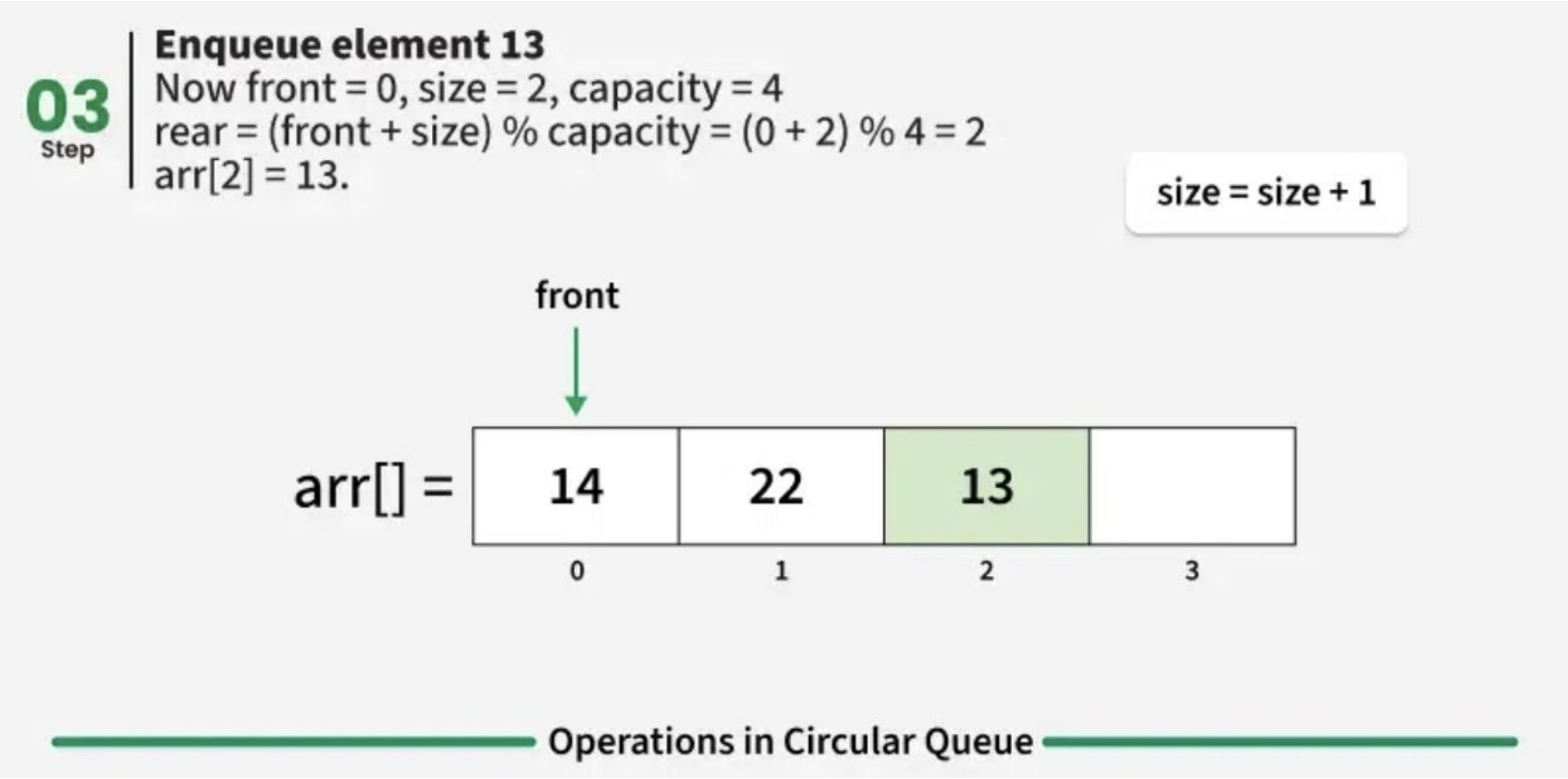
Circular Queue



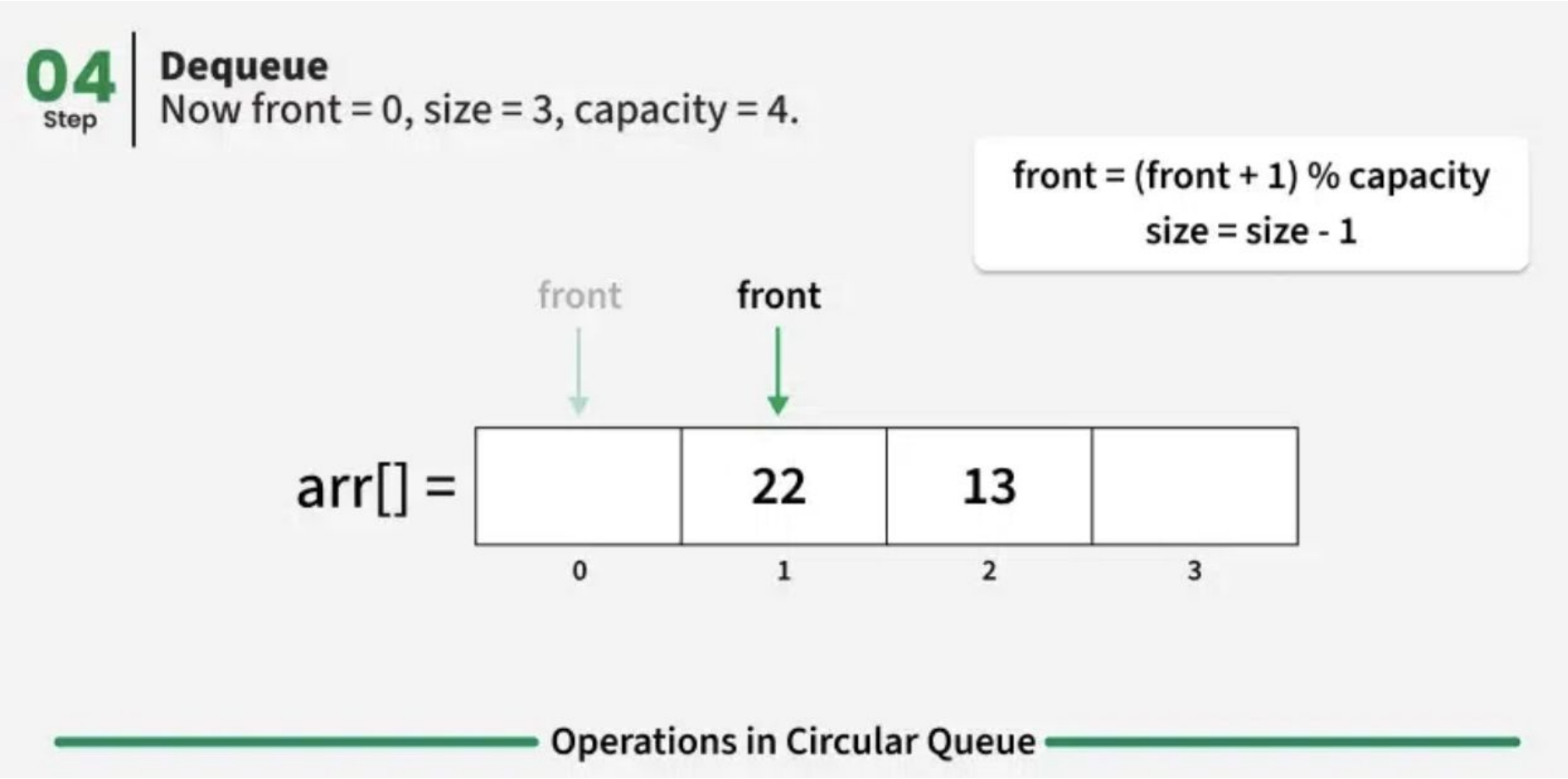
Circular Queue



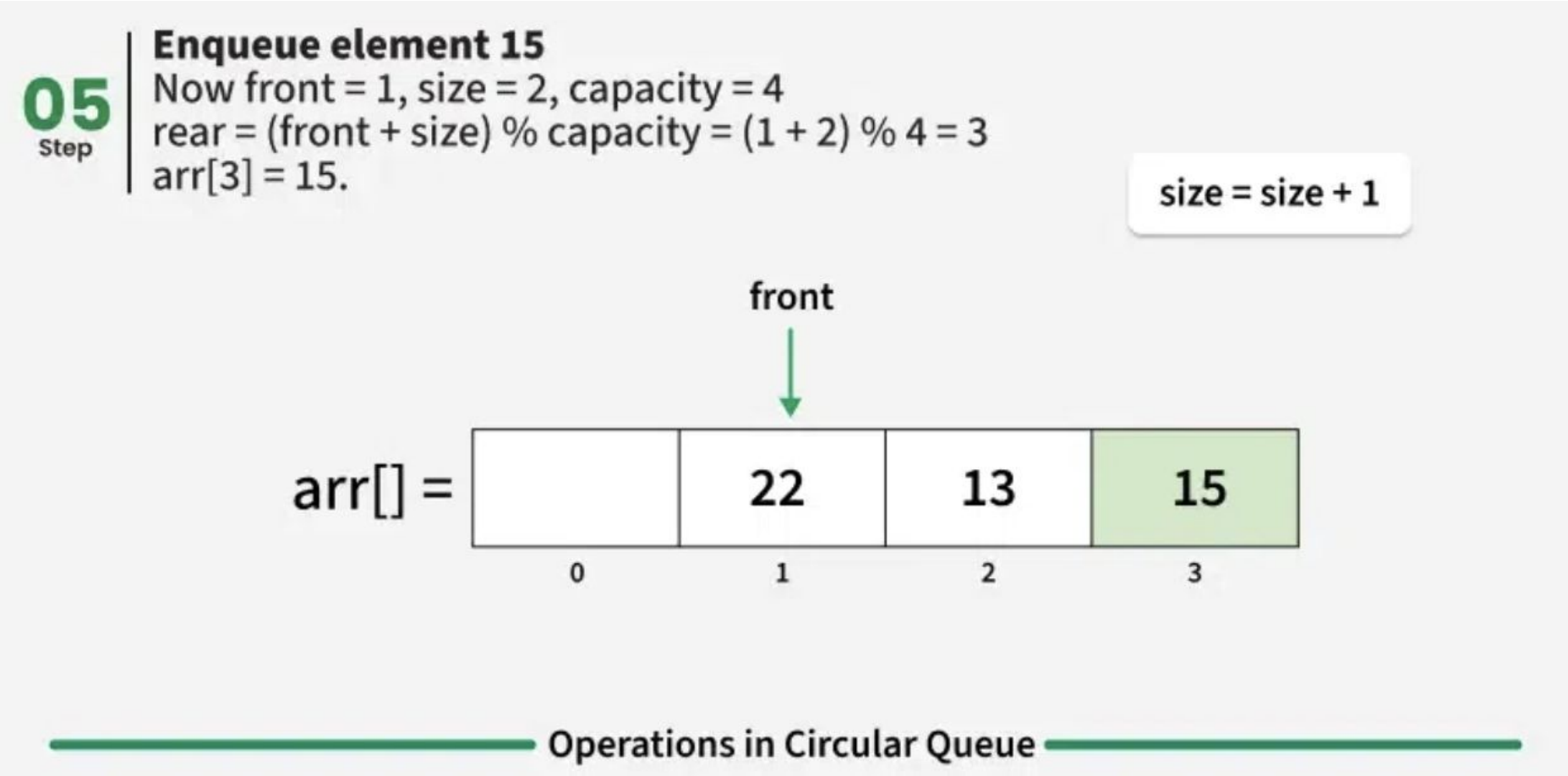
Circular Queue



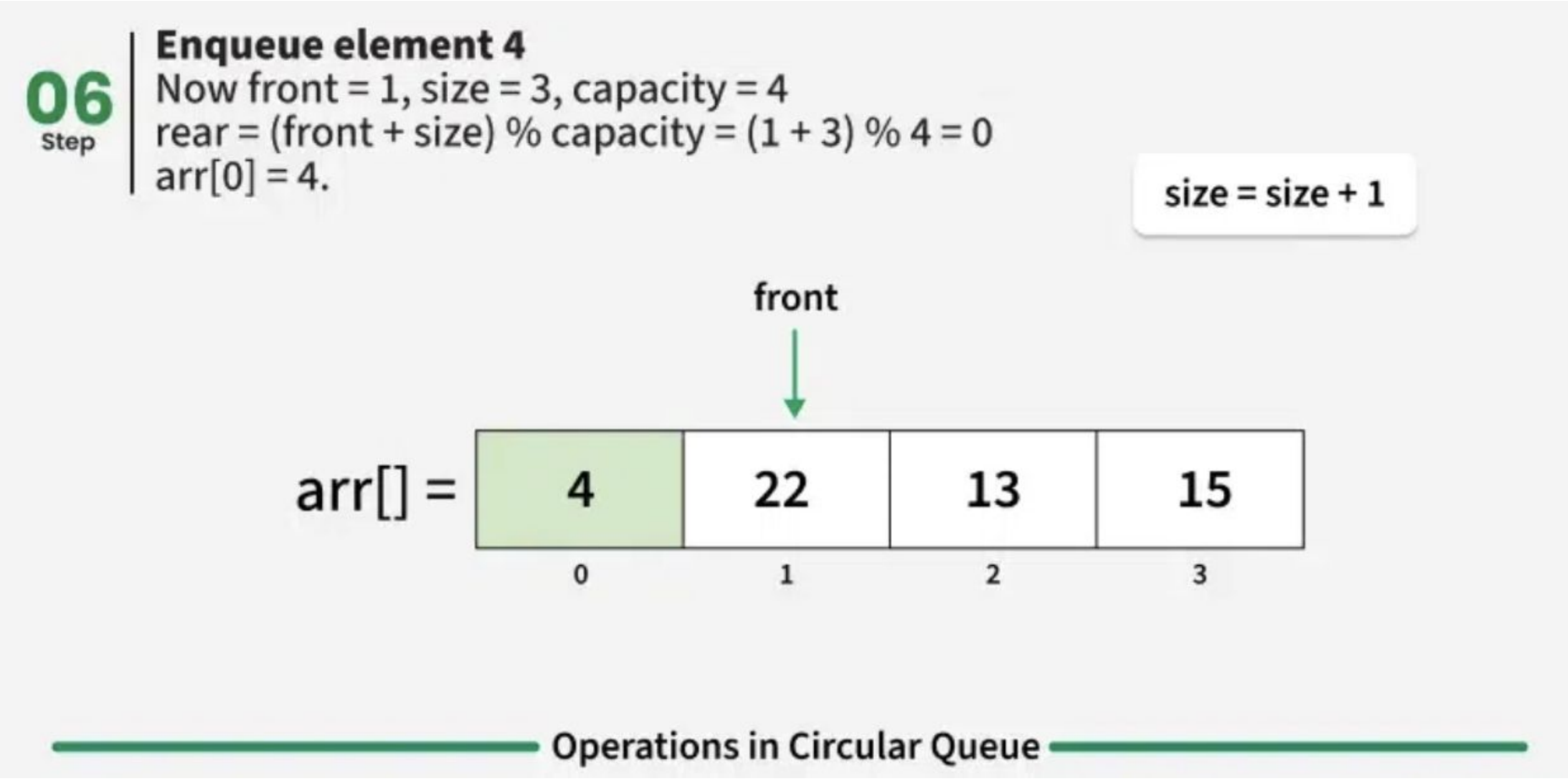
Circular Queue



Circular Queue



Circular Queue

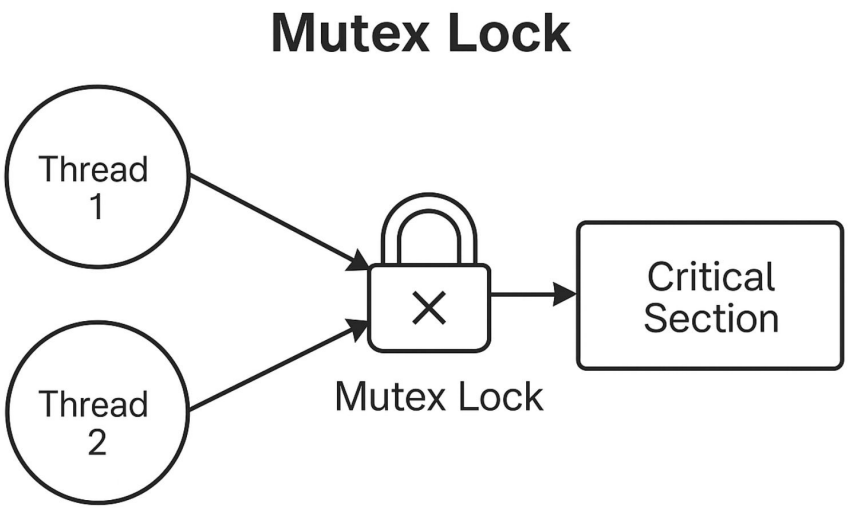


Thread Safe Options

- Mutex Locks
- Semaphores
- Condition variables
- Monitors

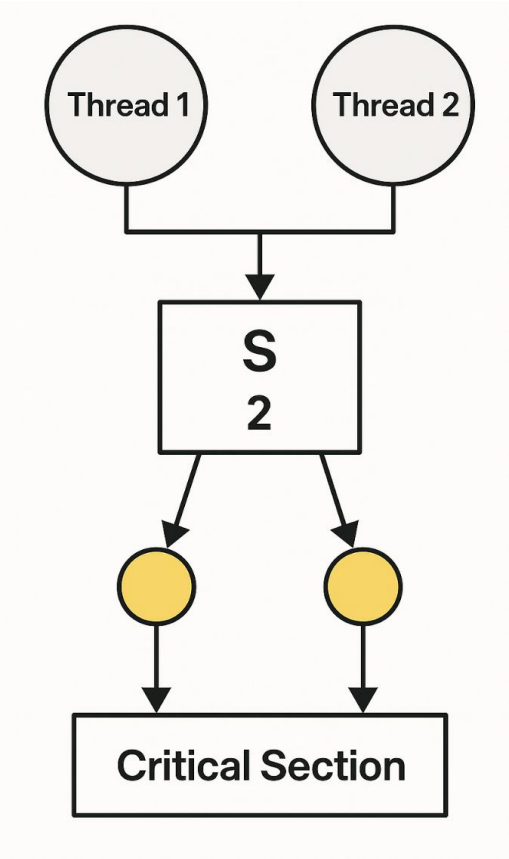
Thread Safe Options

- Mutex Locks
  - Guarantee mutual exclusion
  - Only one thread can hold access at a time
    - This thread must be the one to release the lock
  - Simple and efficient
  - No signaling
- Semaphores
- Condition variables
- Monitors



Thread Safe Options

- Mutex Locks
- Semaphores
  - Binary and Counting Semaphores
    - Only binary ensure mutual exclusion
  - Signaling with wait() and signal()
  - Removes busy waiting
- Condition variables
- Monitors

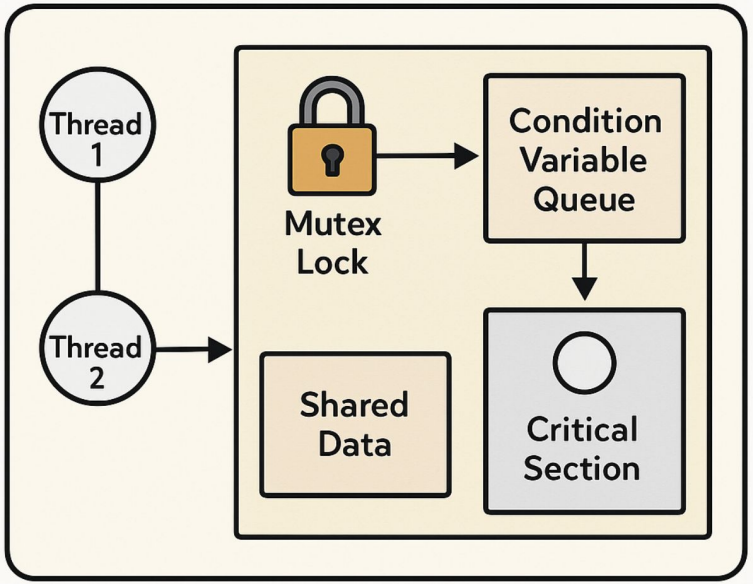


Thread Safe Options

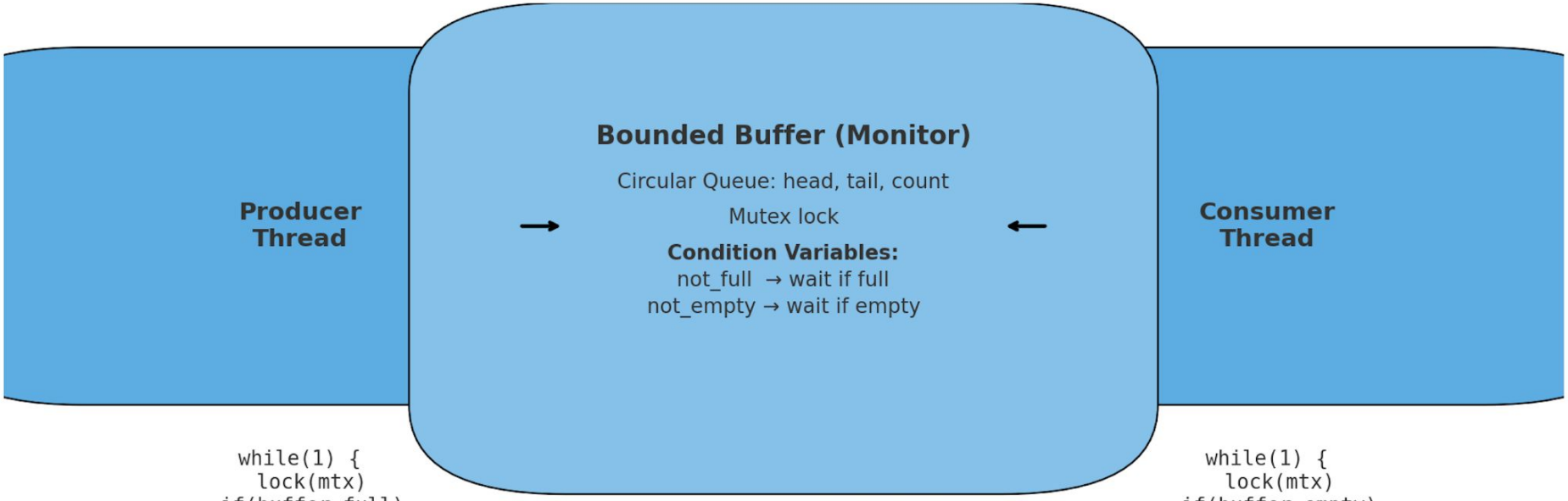
- Mutex Locks
- Semaphores
- Condition variables
  - Allow threads to signal between themselves
    - wait() and signal()
  - No mechanism for mutual exclusion directly
    - Generally combined with mutex locks
  - No resource counting
- Monitors

Thread Safe Options

- Mutex Locks
- Semaphores
- Condition variables
- Monitors
  - Combine mutex locks and condition variables
    - signaling and mutual exclusion
  - Encapsulate:
    - shared data
    - functions which edit data
  - No resource counting



Monitor-Like Approach




```
while(1) {
  lock(mtx)
 if(buffer full)
wait(not_full, mtx);
 put item; count++
 signal(not_empty)
   unlock(mtx)
       }
```

```
while(1) {
  lock(mtx)
 if(buffer empty)
wait(not_empty, mtx);
 get item; count--
 signal(not_full)
   unlock(mtx)
       }
```

Note: wait(cond, mtx) releases the mutex and suspends the thread until signaled. This is NOT busy waiting — the thread sleeps and consumes no CPU while waiting.

DNS Lookup

- Computers use IP Addresses for website lookup while humans use names
- DNS (Domain Name System) is the phonebook of the internet, mapping IP Addresses to names
- www.google.com → 142.250.69.238

DNS Lookup in C

- gethostbyname()
  - older, depreciated
  - works with hostent structs

```
struct hostent {
    char  *h_name;       // Official name of the host
    char **h_aliases;    // Other names (aliases)
    int    h_addrtype;   // Address type
    int    h_length;     // Length of the address
    char **h_addr_list;  // List of IP addresses (binary form)
};
```

DNS Lookup in C

```c
// Lookup the hostname
struct hostent *he = gethostbyname("google.com");

// Check if lookup failed
if (he == NULL) {
    printf("NOT_RESOLVED\n");
    return 1;
}

// Access the official name
printf("Official name: %s\n", he->h_name);

// Get the first IP address and print it
struct in_addr **addr_list = (struct in_addr **)he->h_addr_list;
printf("First IP: %s\n", inet_ntoa(*addr_list[0]));
```

DNS Lookup in C

```c
// Lookup the hostname
struct hostent *he = gethostbyname("google.com");

// Check if lookup failed
if (he == NULL) {
    printf("NOT_RESOLVED\n");
    return 1;
}

// Access the official name
printf("Official name: %s\n", he->h_name);

// Get the first IP address and print it
struct in_addr **addr_list = (
printf("First IP: %s\n", inet_
```

```
Official name: google.com
First IP: 142.250.72.78
```

C Thread Safety Demo

- Download it here!